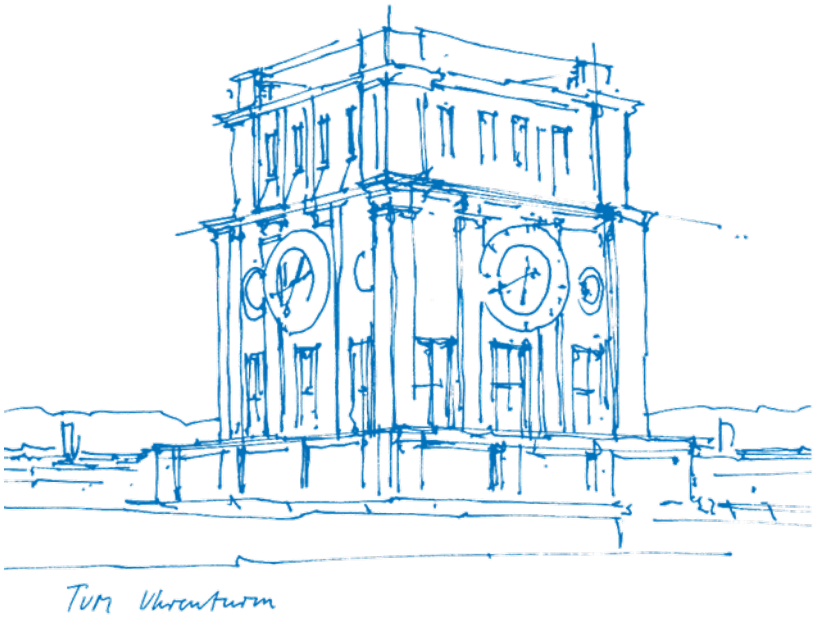
Coupled simulations with OpenFOAM and other solvers through the preCICE coupling library

Gerasimos Chourdakis

Technische Universität München
Fakultät für Informatik
Informatik 5 – Wissenschaftliches Rechnen

School of Chemical Engineering
National Technical University of Athens
January 9, 2018



Contents

Contents

Contents

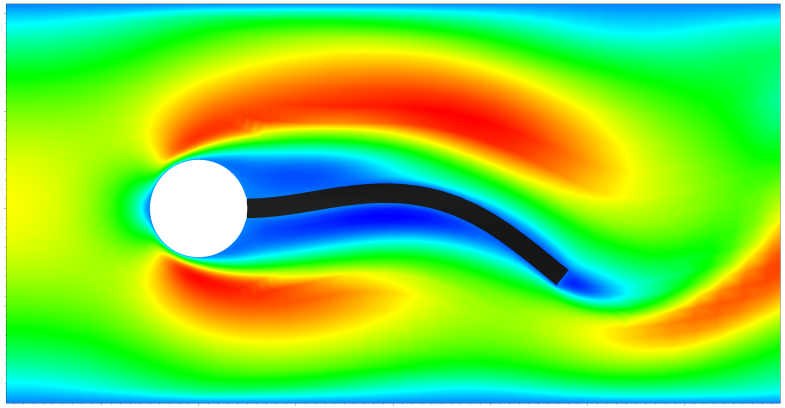
I. preCICE and OpenFOAM: 15min + 5min discussion

- Multi-physics simulations
- preCICE
- OpenFOAM
- The OpenFOAM adapter
- Simulation of a shell-and-tube heat exchanger

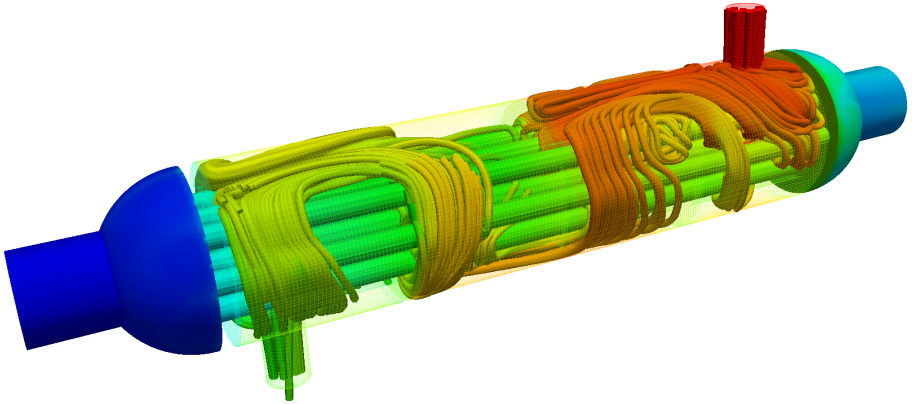
II. Study in TUM: 15min + 25min discussion

- Cool in TUM
- Degree Programs
- How to apply

Multi-physics simulations

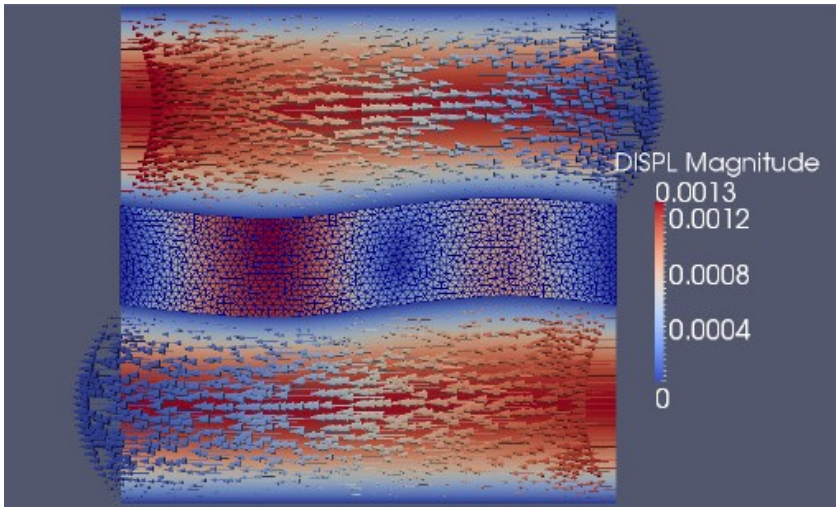


Fluid-Structure Interaction: Flow around an elastic flap

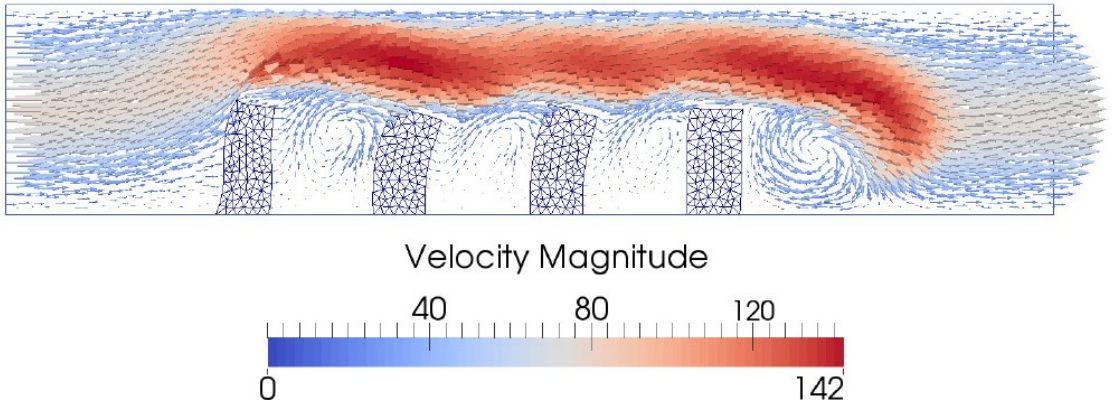


Conjugate Heat Transfer: Shell-and-tube heat exchanger

Multi-physics simulations



Fluid-Structure Interaction: flow around a flexible membrane

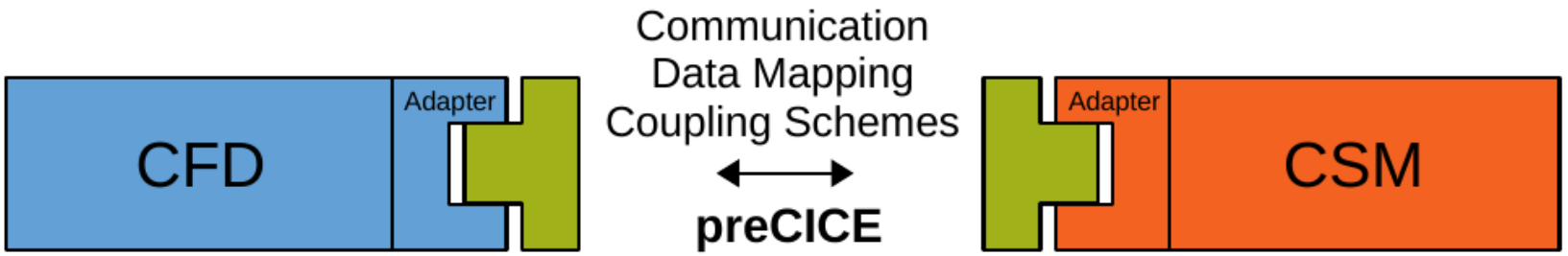


Fluid-Structure Interaction: flow above multiple beam-like structure

preCICE

Precise Code Interaction Coupling Environment

$$\text{solver}_\text{A} + \text{preCICE} + \text{solver}_\text{B} \rightarrow \text{multiphysics}$$



- Free (GNU LGPL), developed at the Technical University of Munich and the University of Stuttgart.
- C++ library – API in C, C++, Fortran, Python
- Official adapters for CalculiX, Code_Aster, COMSOL, Fluent, OpenFOAM, SU2, ...

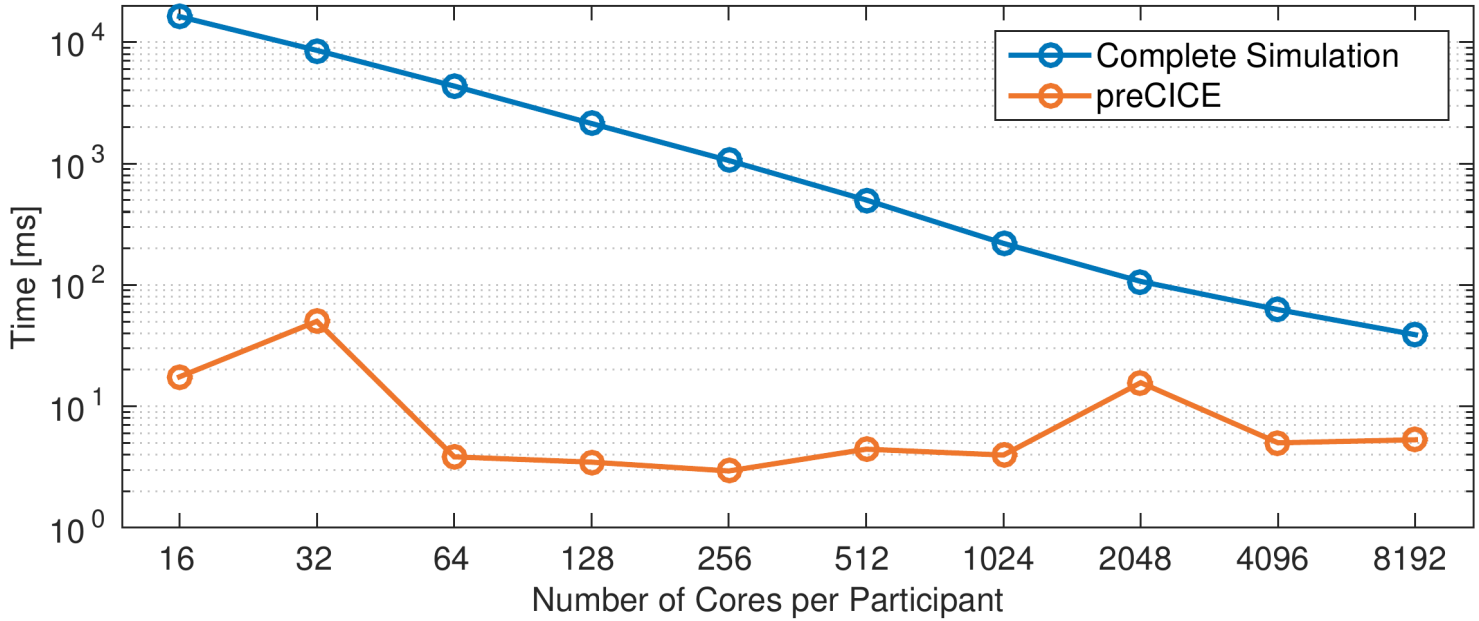
Communication

Asynchronous communication via MPI or TCP/IP sockets

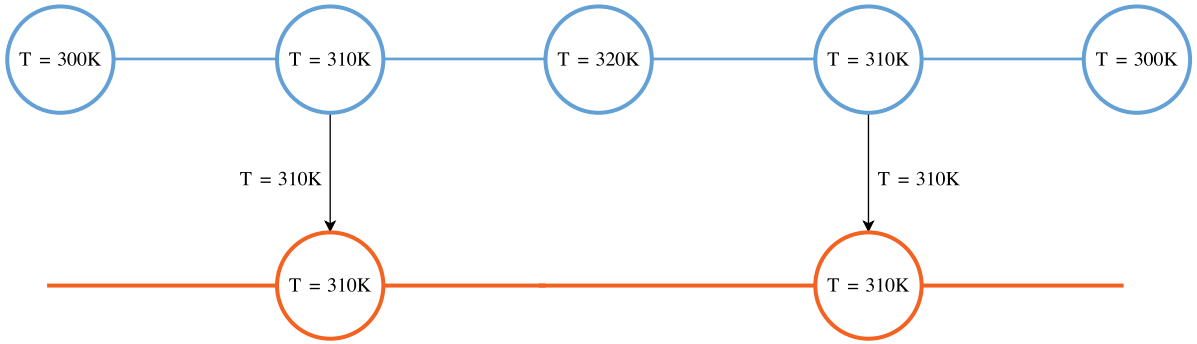
Special in preCICE:

library approach: the coupled solvers load the preCICE library and communicate directly

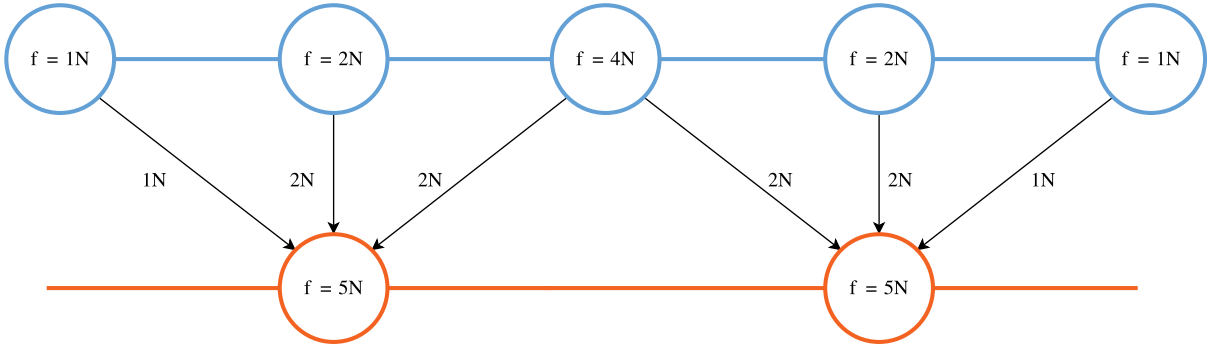
fully parallel, peer-to-peer: each process can communicate with any other process directly



Mapping between meshes



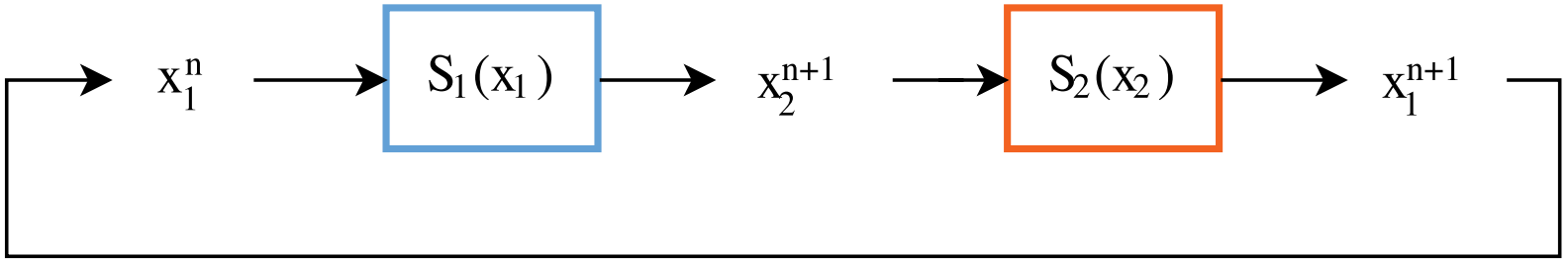
Consistent mapping example (temperatures)



Conservative mapping example (forces)

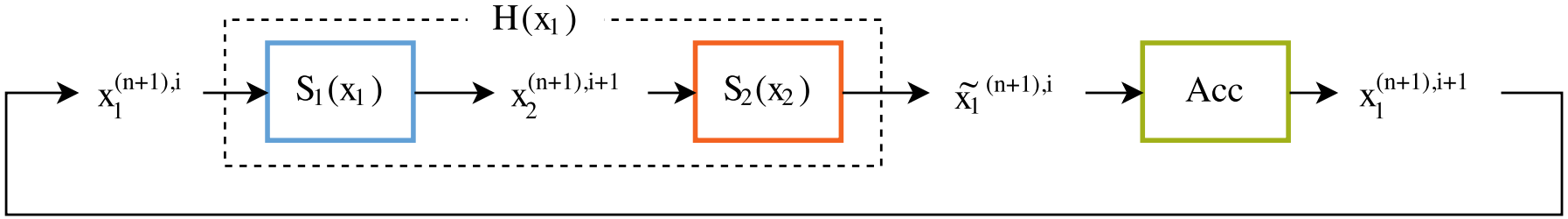
See also: nearest-neighbor / nearest-projection / radial-basis functions (RBF) mapping

Explicit & implicit coupling



$$n \leftarrow n + 1$$

Serial-explicit scheme. A parallel version also exists.

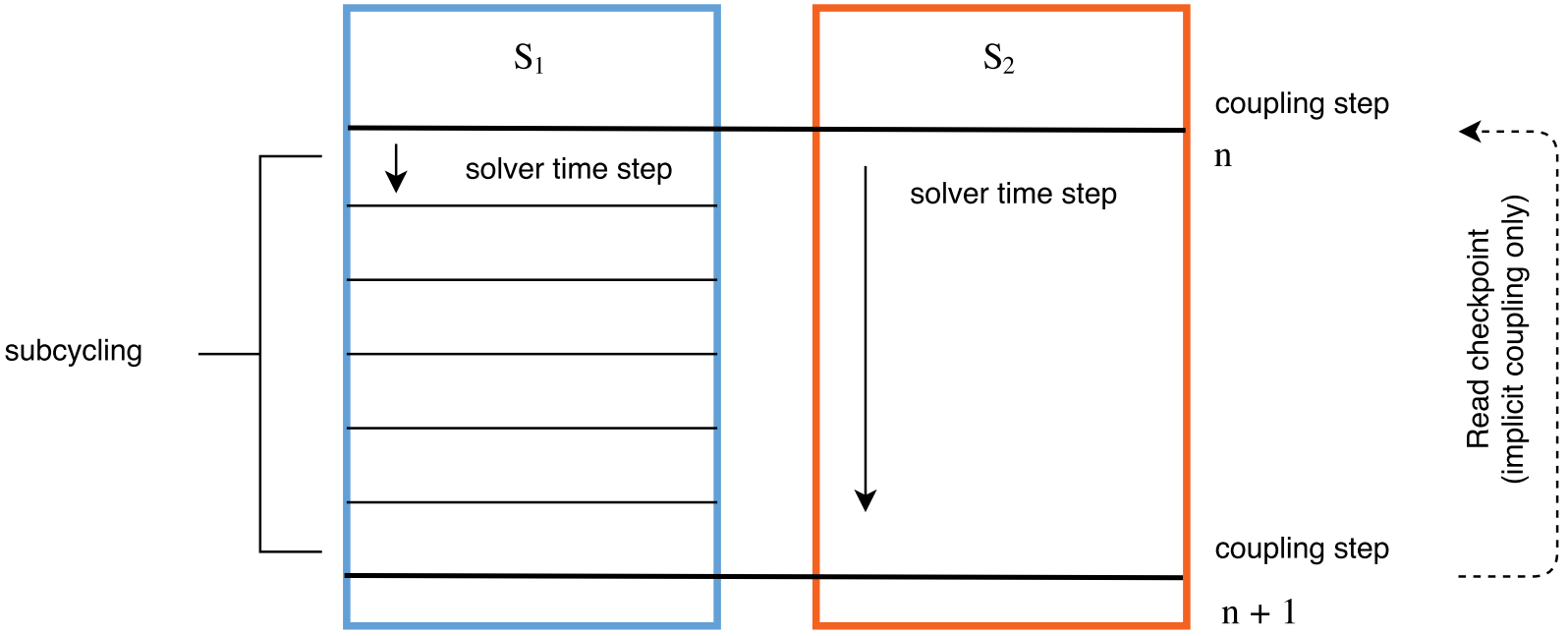


$$i \leftarrow i + 1, \quad S_1 \leftarrow S_1^{(n)}, \quad S_2 \leftarrow S_2^{(n)}$$

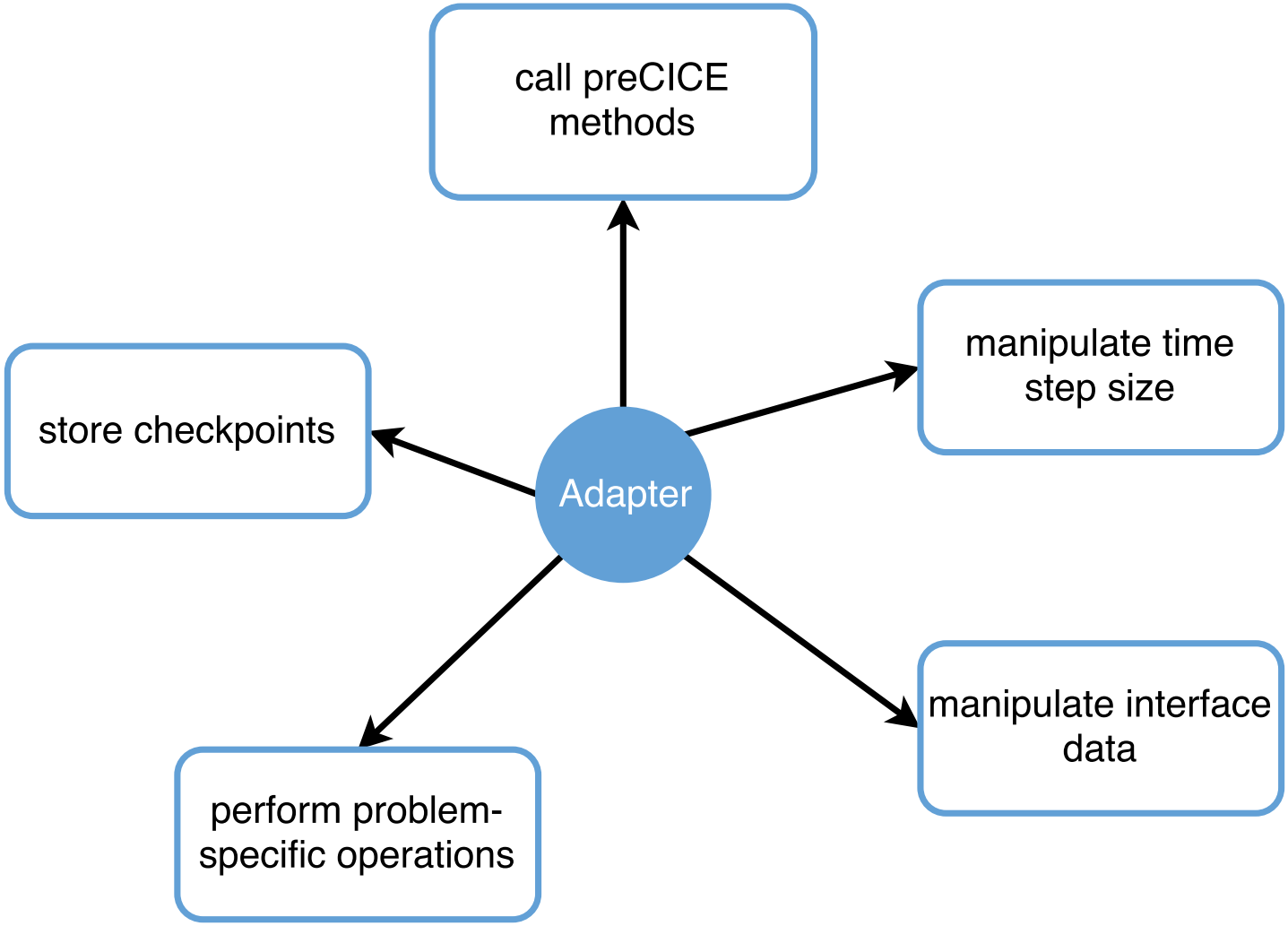
Serial-implicit scheme. A parallel version also exists. When the solution converges, increase *n*.

Checkpointing

Reload the state of the solver from the last completed coupling time step after every non-converged implicit coupling iteration:



The roles of an adapter

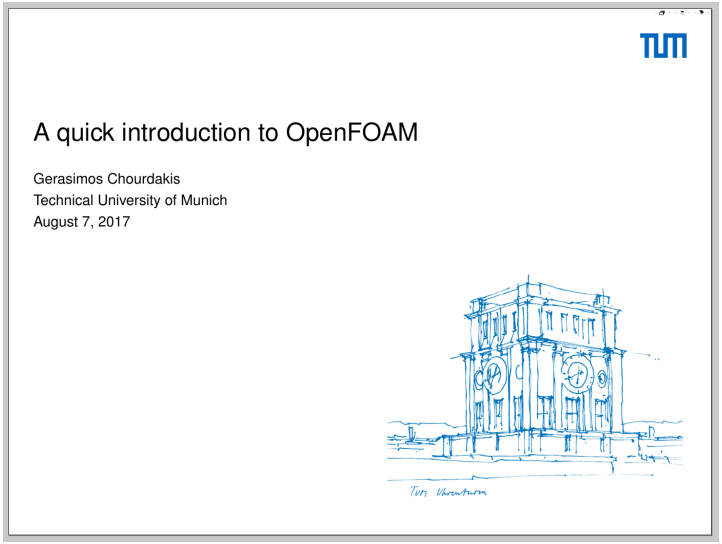


OpenFOAM

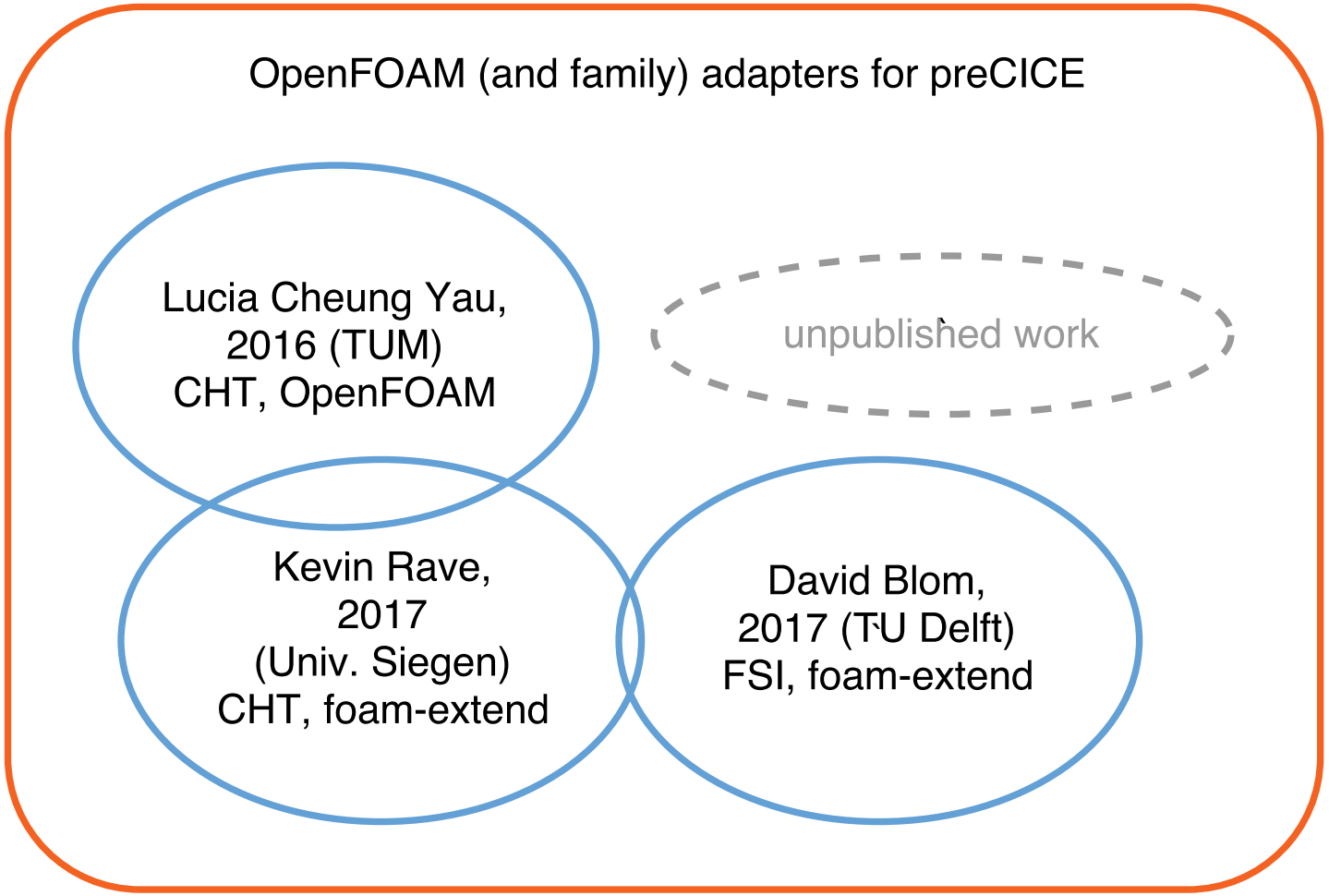
Open-source Field Operation And Manipulation

- **Collection** of tools for continuum mechanics (mainly CFD)
- **Framework** for in-house solvers
- **Several variants:** OpenFOAM (openfoam.org), OpenFOAM+ (openfoam.com), foam-extend
- Free (GNU GPL), C++

Want to learn more about OpenFOAM? Ask for material!



Duplicated development effort



All these adapters are bound to specific solvers.

Example of an adapted solver (previous)

```
while (adapter.isCouplingOngoing()) {
  #include "readTimeControls.H"
  #include "compressibleCourantNo.H"
  #include "setDeltaT.H"

  /* Adapter: Adjust solver time */
  adapter.adjustSolverTimeStep();

  /* Adapter: Write checkpoint */
  if(adapter.isWriteCheckptReq())
  {
    adapter.writeCheckpoint();
    adapter.fulfilledWriteCheckpt();
  }

  runTime++;

  /* Adapter: Receive coupling data */
  adapter.readCouplingData();

  /* solve equations (not shown) */

  /* Adapter: Send and advance */
  adapter.writeCouplingData();
  adapter.advance();

  /* Adapter: Read checkpoint */
  if(adapter.isReadCheckptRequired())
  {
      adapter.readCheckpoint();
      adapter.fulfilledReadCheckpt();
  }

  if(adapter.isCouplTimeStepComplete())
    runTime.write();
}
```

Before: A working and validated prototype

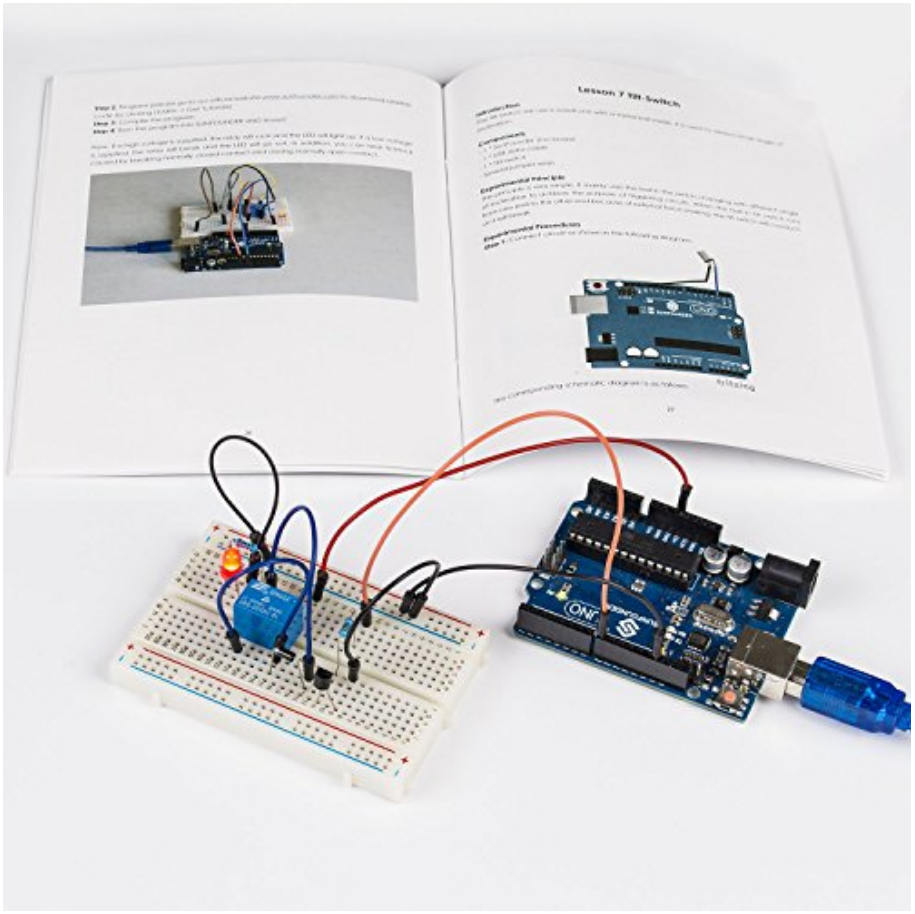
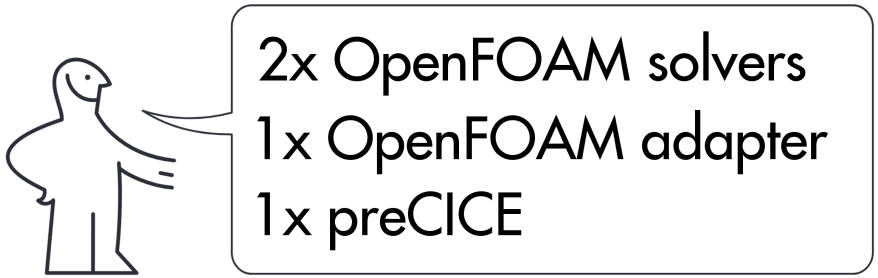


Image from desertcart.ae.

Goal: A user-friendly, plug-and-play adapter

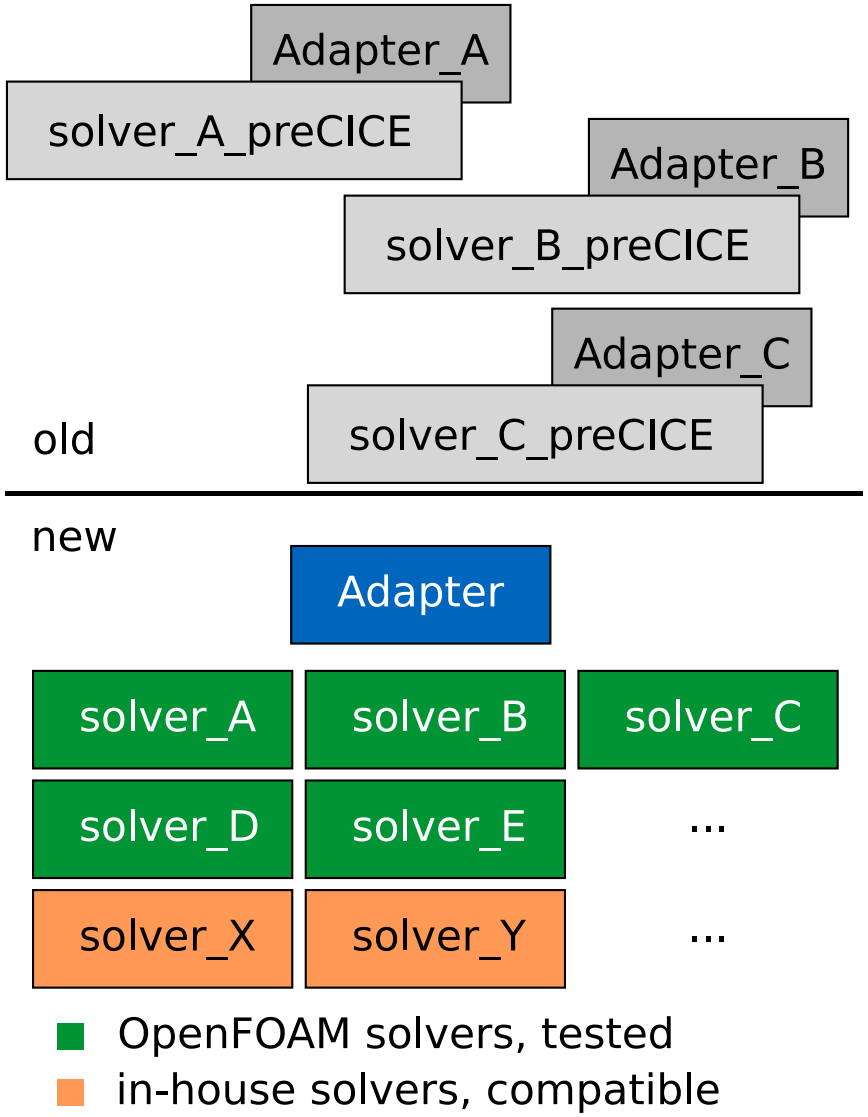
MULTIFYSIK



The human-like figure is a property of ikea.com.

Issues

1. Changes in the solvers required
   - Effort to adapt
   - Recompilation
   - Maintenance
2. Solver-specific adapters required
   - Variety of adapted solvers limited
   - Maintenance
3. OpenFOAM versions compatibility
   - Adapted solvers bound to version
   - Limited user base



OpenFOAM configuration

```
// system/controlDict
functions
{
    preCICE_Adapter
    {
        type preciceAdapterFunctionObject;
        libs ("libpreciceAdapterFunctionObject.so");
    }
}
```

Set the boundary condition types:

```
// 0/T
interface
{
    type            fixedValue;
    value           uniform 300;
}

// other types: fixedGradient, mixed
```

Properties for incompressible solvers:

```
// constant/transportProperties
rho     rho [ 1 -3  0  0 0 0 0 ] 1;
Cp      Cp  [ 0  2 -2 -1 0 0 0 ] 5000;
```

Properties for basic solvers:

```
// constant/transportProperties
k       k   [ 1  1 -3 -1 0 0 0 ] 100;
```

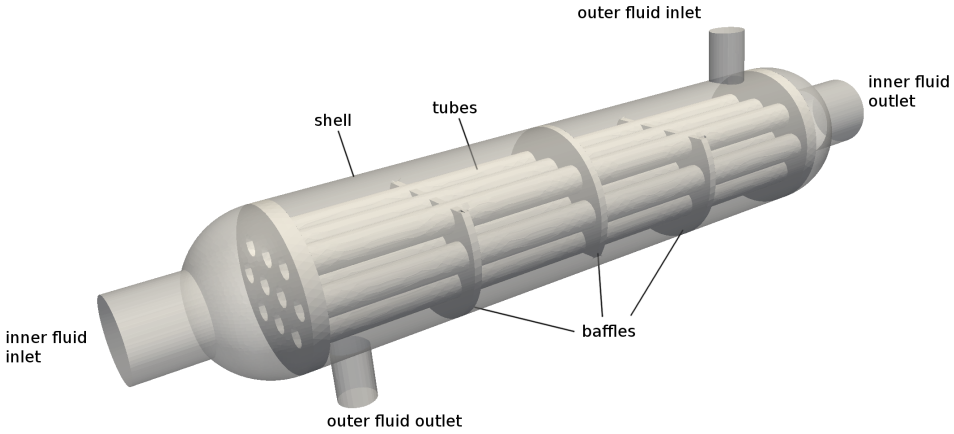
Adapter's configuration file

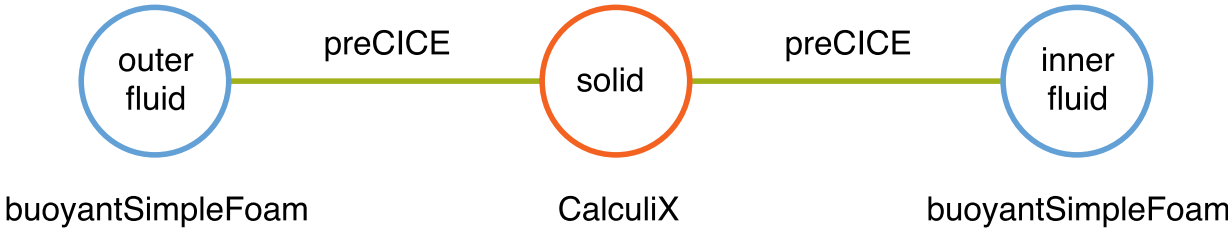
```
participant: Fluid

precice-config-file: precice-config.xml

interfaces:
- mesh: Fluid-Mesh
  patches: [interface]
  write-data: Temperature
  read-data: Heat-Flux
```

In-house solver with different field names?
No problem!

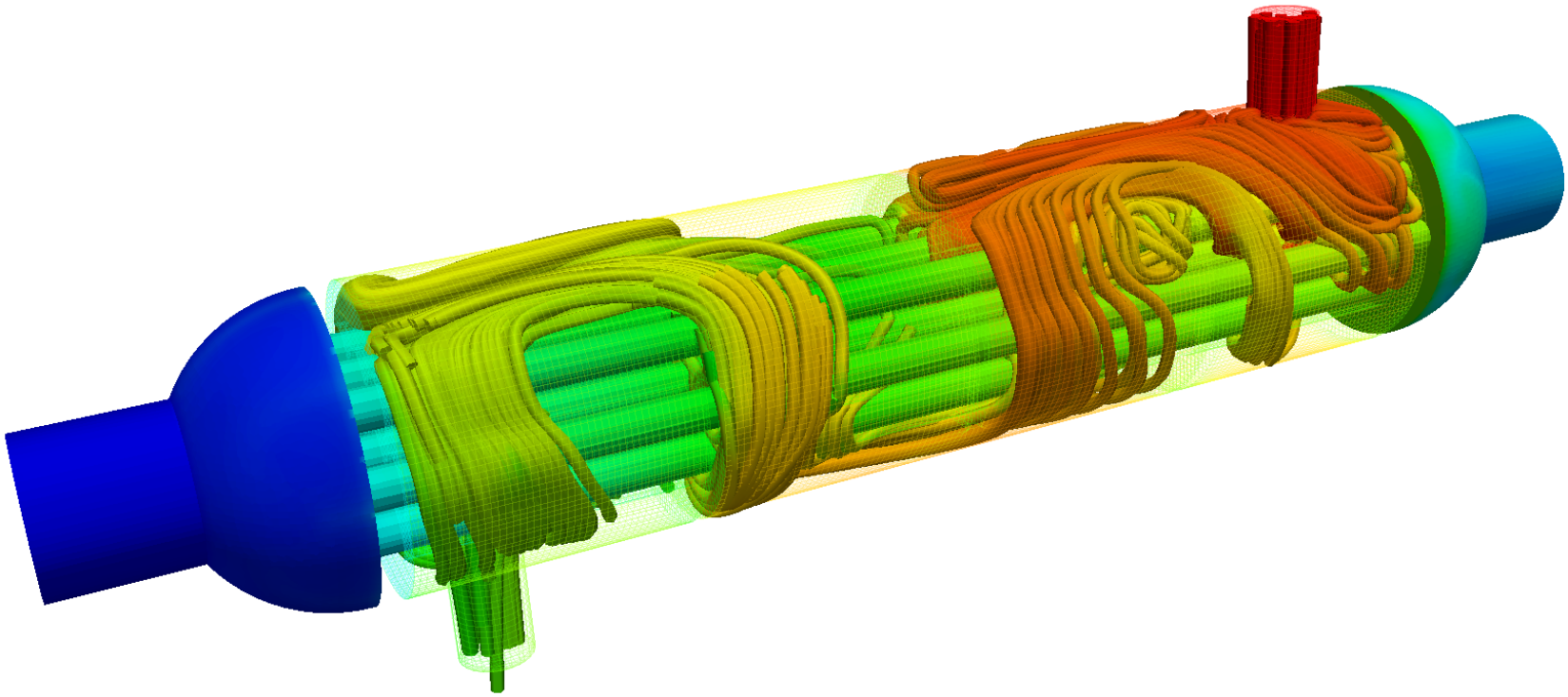
```
# Temperature field
nameT: T
# transportProperties dictionary
nameTransportProperties: transportProperties
# thermal conductivity
nameKappa: k
# density
nameRho: rho
# heat capacity for constant pressure
nameCp: Cp
# Prandtl number
namePr: Pr
# turbulent thermal diffusivity
nameAlphat: alphat

# user-set solver type
# (overrides the auto-determined)
solverType: compressible
```

Simulation of a shell-and-tube heat exchanger



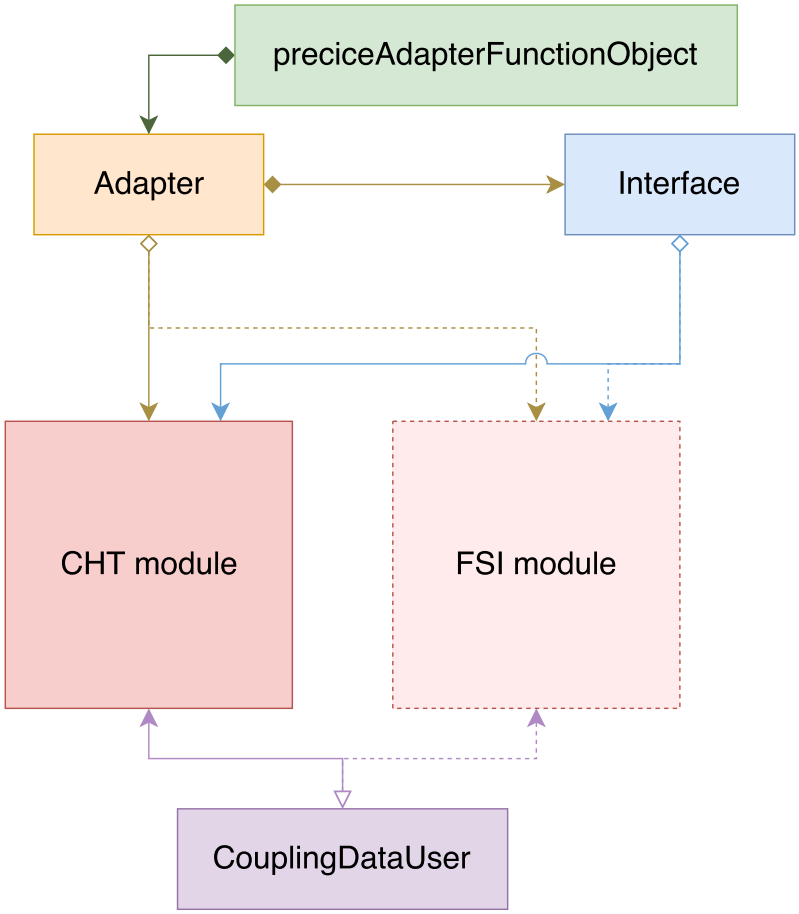


Simulation of a shell-and-tube heat exchanger



Steady-state / explicit Robin – Robin coupling: results identical for every write-time.

An extensible adapter



Questions on preCICE?

preCICE

Website: `precice.org`
preCICE is on GitHub: `github.com/precice`

Contents

I. preCICE and OpenFOAM: 15min + 5min discussion

- Multi-physics simulations
- preCICE
- OpenFOAM
- The OpenFOAM adapter
- Simulation of a shell-and-tube heat exchanger

II. Study in TUM: 15min + 25min discussion

- Cool in TUM
- Degree Programs
- How to apply

Cool in TUM

See photos

Degree programs

`tum.de` → `en` → “Studies”: “Degree Programs”

- Computational Science and Engineering (EN, winter)
- Computational Mechanics (EN, winter)
- Chemical Engineering (DE, winter/summer)
- Brewing and Beverage Technology (DE, winter)
- Industrial Chemistry (EN, deadline: 31/3, Singapore)
- Industrial Biotechnology (DE & EN, winter/summer)
- Biomass Technology (EN & DE, winter)
- Biochemistry (DE, winter/summer)
- Food Technology and Biotechnology (DE, winter)
- Food Chemistry (DE, winter)
- Aerospace (DE, winter/summer)
- Automotive and Combustion Engine Technology (DE, winter/summer)
- Energy and Process Engineering (DE, winter/summer)
- Power Engineering (EN, winter)
- Environmental Engineering (EN, winter/summer)
- Renewable Resources (DE, winter/summer)
- ...

M.Sc. Computational Science & Eng.

`cse.tum.de`

- Required modules
  - Computer Science (required part) - 10 ECTS
    - Advanced Programming
    - Parallel Programming
  - Numerical Analysis - 21 ECTS
    - Numerical Programming I & II
    - Parallel Numerics
  - Scientific Computing - 21 ECTS
    - Scientific Computing Lab
    - Scientific Computing I & II
    - Seminar: Scientific Computing
- Elective modules - 15 + 23 ECTS
- Master's Thesis - 30 ECTS

M.Sc. Computational Science & Eng.

`cse.tum.de`

- Elective modules
  - Computer Science (elective part)
    - Computer Architecture and Networks
    - Fundamental Algorithms
    - Patterns in Software Engineering
    - Scientific Visualization
    - Programming of Supercomputers
  - Applications of CSE (catalogs)
    - Computational Mechanics
    - Computational Fluid Dynamics
    - Mathematics in Bioscience
    - Computational Physics
    - Computational Electronics
    - Computational Chemistry
  - Methods and Techniques of CSE (catalogs)
    - Algorithms in Scientific Computing
    - Finite Elements
    - Parallel and Distributed Systems, HPC
    - Computational Visualization
    - Computational Stochastics and Statistics
    - Big Data

How to apply

- Application deadlines:
  - Summer semester: January 15
  - Winter semester: May 31
  - DAAD scholarships: usually in November
- Documents:
  - Online application
  - CV, Letter of Motivation
  - Two letters of Recommendation
  - English: TOEFL / IELTS / Cambridge
  - Diploma, Grades
- **Submit notarized copies!** Best way: go to the German embassy
- Help & Scholarships: `daad.gr`
- More help: Greeks@TUM - `greeks.fs.tum.de` (also in Facebook)

Questions?

e-mail: gerasimos.chourdakis@tum.de